SEMANTICOPT: TOWARDS LLM-BASED SEMANTIC BLACK-BOX OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimizing expensive black-box systems often requires both domain knowledge and effective sequential decision making. Bayesian optimization (BO) provides strong numerical refinement and uncertainty-aware search, but typically cannot directly use broader semantic information such as expert knowledge, scientific documentation, or prior experiments. Large language models (LLMs) can exploit such information to propose strong initial configurations, but their optimization performance often plateaus as numerical observations accumulate. We introduce SemanticOpt, an LLM-based optimizer that learns model-based optimization behavior from BO trajectories. We construct more than 100,000 optimization trajectories combining context-informed LLM initialization with multiple BO methods across diverse synthetic and real-world-like function spaces, and fine-tune an LLM to jointly propose candidate configurations and predict objective value, uncertainty, and failure probability. Across 42 scientific and engineering black-box optimization problems, SemanticOpt achieves a strong semantic initialization relative to BO while substantially improving iterative refinement over longer optimization trajectories. Its predictions also become more accurate and better calibrated as observations accumulate, and the model is more robust than the base LLM to misleading semantic information. These results suggest that supervision from numerical optimizers can teach LLMs to better incorporate accumulating numerical evidence while preserving their ability to exploit natural-language priors.

1 INTRODUCTION

In many scientific and engineering workflows, researchers must optimize the performance of a system while only being able to run a small number of costly evaluations. By testing different parameter settings, the user iteratively builds an understanding of achievable performance and decides which experiment to perform next. Each evaluation may involve training a large model, running an expensive simulation, or performing a physical experiment, and the relationship between the design parameters and the final outcome is typically unknown.

To illustrate this type of problem, consider optimizing a chemical reaction process to produce a desired compound. A scientist may want to maximize product yield by adjusting variables such as temperature, pH, and reaction time. Each trial may require running an experiment for several days before measuring the final concentration, so a limited number of experiments can be performed. A standard black-box optimizer, such as Bayesian optimization (BO), only observes the numerical configurations and the measured yield. It does not know that high temperatures may accelerate reaction rates but increase byproducts. Even if a numerical optimizer observes one high-temperature trial with low yield, it cannot determine whether high temperature is generally harmful or helpful only at a different pH or reaction time. An effective semantic optimizer would use both numerical history and semantic context to select better reaction conditions with a limited evaluation budget while also providing reliable surrogate-like predictions for each selected configuration.

Large language models (LLMs) offer a powerful mechanism for handling contextual information but are not designed for reliable iterative optimization. Taking advantage of their knowledge base, LLMs are often capable of successful warm-starting (proposing initial experiments). However, performance often plateaus over longer trajectories. Recent studies have shown that LLMs can be effective during the initialization stage of search, but alternative methods or a combination of LLM

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

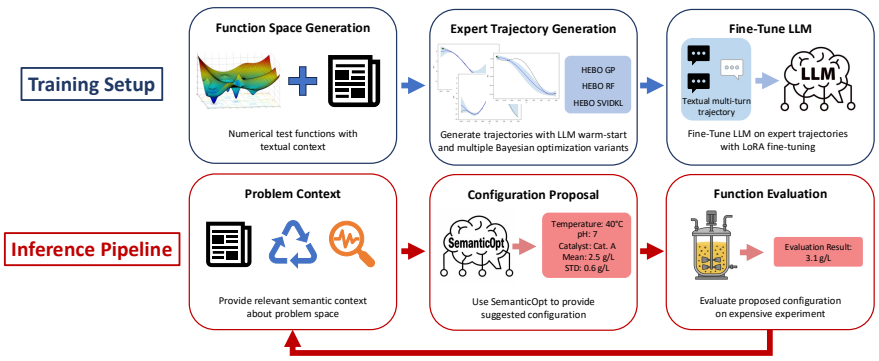


Figure 1: Overview of SemanticOpt training and inference.

guidance with BO methods are needed for LLMs to be effective iterative optimizers (Liu et al., 2024; Yuan et al., 2026; Huang et al., 2024; Ferreira et al., 2026). This suggests that while LLMs contain useful priors for proposing strong initial candidates, they lack the exploration-exploitation balance and uncertainty-aware adaptation that makes BO effective over extended optimization horizons.

We show that this missing behavior can be learned directly from BO. We introduce SemanticOpt, which fine-tunes an LLM on structured optimization trajectories generated by a combination of LLM initialization and multiple BO methods on a large synthetic dataset. Through this supervision, the LLM learns to keep refining its search as observations accumulate and to predict outcomes and uncertainty similar to the BO surrogates, while preserving its ability to exploit semantic context and learning to revise its decisions when observations disagree. Our results suggest that trajectories from reliable numerical optimizers are a valuable source of supervision for teaching language models sequential decision-making under evidence. Our key contributions are as follows:

1. We construct a training dataset of more than 100,000 optimization trajectories over diverse GP-generated and LLM-generated function spaces with semantic context, using multiple BO variants to provide varied optimization behavior.
2. We introduce SemanticOpt, which fine-tunes an LLM to jointly propose configurations and predict objective value, uncertainty, and failure probability, learning from both the decisions and the surrogate predictions of its BO teachers.
3. We show that this supervision lets the LLM refine on par with its BO teachers and predict outcomes with uncertainty comparable to their surrogates. It also keeps a large early advantage over BO from semantic context and is substantially more robust than the base model to misleading context.

2 Related Work

Bayesian Optimization (BO) BO is a widely used global optimization framework for expensive black-box functions because it can achieve strong sample efficiency by balancing exploration and exploitation through acquisition functions such as Log Expected Improvement (Ament et al., 2023). Additionally, BO can utilize different surrogate models, such as Gaussian processes (GPs), which may perform better in different types of landscapes and parameter spaces. BO has become broadly accessible through open-source tools including Spearmint (Snoek et al., 2012), BoTorch (Balandat et al., 2020), AutoOED (Tian et al., 2021), SMAC3 (Lindauer et al., 2022), HEBO (Cowen-Rivers et al., 2022), Openbox (Jiang et al., 2024), and Vizier (Golovin et al., 2017; Song et al., 2024b). In particular, HEBO (Cowen-Rivers et al., 2022) has emerged as a high-quality and competitive optimizer, demonstrating strong empirical performance across benchmark suites (Turner et al., 2021).

Learning to Optimize Learned black-box optimization has progressed from early RNN-based optimizers (Li & Malik, 2016; Andrychowicz et al., 2016) to transformer-based approaches enabled by larger datasets and increased compute. Early transformer methods explored online RL for task-specific solvers, including MELBA (Chaybouti et al., 2022) and Neural Acquisition Process (Mar-

aval et al., 2024), while POM (Li et al., 2024) targeted high-dimensional evolutionary search. More recent work emphasizes broader generalization: PFNs4BO (Müller et al., 2023) learns surrogate-model behavior, OptFormer (Chen et al., 2022) trains text-based transformers on proprietary optimization data, RIBBO (Song et al., 2024a) uses offline reinforcement learning with conditional rewards, and ZeroShotOpt (Meindl et al., 2025) shows that transformers trained on large synthetic function sets can imitate BO trajectories. These results suggest that large-scale training can produce optimization policies with cross-task generalization.

LLM-based Optimization Large language models have also been used directly for black-box optimization. LLAMBO (Liu et al., 2024), Reasoning BO (Yang et al., 2025), and EvoLLM (Lange et al., 2024) show that prompting strategies, iterative interaction, and contextual information can produce competitive optimization trajectories in scenarios where LLM knowledge is strong, such as machine learning hyperparameter optimization. Additionally, LGBO (Yuan et al., 2026) and LLINBO (Chang et al., 2025) provide frameworks for integrating LLM outputs with BO acquisition functions, representing another path toward incorporating semantic information into model-based decision making. These methods highlight the promise of LLMs for using semantic information in optimization, but use the LLM as an auxiliary component in a BO pipeline or require fitting a GP in addition to LLM outputs. A related line of work uses language models within the surrogate itself. Kristiadi et al. (2024) investigate fine-tuned LLM representations and probabilistic uncertainty estimation for molecular optimization, while GOLLuM (Ranković et al., 2026) jointly adapts an LLM representation and GP through the GP marginal likelihood, producing task-adapted representations with calibrated predictive uncertainty. In contrast, our work fine-tunes an LLM directly on structured BO trajectories with semantic context, training it to produce both candidate configurations and surrogate-style predictions with uncertainty within a single end-to-end optimizer.

Overall, prior work suggests three complementary strengths: BO provides reliable sample-efficient search, learned optimizers can emulate desired optimization behavior, and LLMs can utilize semantic information that standard numerical optimizers ignore. SemanticOpt combines these strengths by fine-tuning an LLM on large-scale BO trajectories paired with natural-language problem context and surrogate-model predictions, creating an end-to-end semantic optimizer.

3 METHODOLOGY

Problem Framing We cast optimization as a sequential decision-making problem in which an LLM policy π_θ interacts with an unknown objective function f through multi-turn prompting. Each evaluation queries a configuration $\mathbf{x}_i \in \mathcal{X}$, where $\mathcal{X}$ may contain continuous, integer, or categorical parameters, and returns a scalar objective value $y_i = f(\mathbf{x}_i)$. Each prompt also includes semantic context C, such as problem descriptions, historical experiments, or evaluation-time feedback. At step t, the LLM generates the next configuration given the observation history and semantic context:

$$\mathcal{D}_t = \{(\mathbf{x}_i, y_i)\}_{i=1}^{t}, \qquad \mathbf{x}_{t+1} \sim \pi_\theta(\cdot \mid C, \mathcal{D}_t).$$

The generated configuration $\mathbf{x}_{t+1}$ is then evaluated by the objective function and appended to the history. The goal is to achieve the best configuration within the evaluation budget.

3.1 FUNCTION SPACE GENERATION

Because widely available real-world benchmark functions are limited, we construct a large-scale training dataset using realistic and diverse synthetic objectives. We generate two classes of evaluation functions for this purpose: Gaussian process (GP)-based functions and LLM-generated function spaces. Together, these sources provide both scalable synthetic diversity and semantically grounded problem structures. GPs serve as flexible function generators that represent a wide variety of complex functions, enabling large-scale data collection while producing rich optimization landscapes. The motivation is that GP-based Bayesian optimization methods are most widely used as they perform well across many function classes (Shahriari et al., 2015; Wang et al., 2023). Therefore, by training on GP-generated landscapes, our objective is to learn a policy that can be similarly generalized in diverse problem spaces. We augment sampled functions with nonlinear coordinate warping, discontinuities, and failure regions across the landscape.

Vertical Farm Spectrum Optimization

Task: Design an LED lighting recipe for indoor crop growth.
Inputs: Spectrum ratios, PPFD target, photoperiod length, light cycle type, and far-red pulse use.
Extra Evaluation Context: Each trial also returns stem height index, canopy penetration efficiency, and total energy cost.
Objective: Maximize biomass yield efficiency.

Dynamic Shipping Time Calibration

Task: Tune a shipping policy to reduce average delivery time.
Inputs: Free-shipping limit, express cost, and max wait time before switching carriers.
Historical Context: Other markets found that a medium express cost, high free-shipping limit and long switch-wait lowered delivery time by 18%.
Objective: Minimize average shipping time.

Figure 2: Example problems with extra evaluation metrics and historical context from the generated training dataset. This context is used to begin trajectories with LLM-generated initializations before switching to HEBO methods to generate high-quality examples for LLM fine-tuning.

We also use LLM-generated function spaces to introduce real-world semantic context into the dataset. To do this, we prompt Qwen 3.5 35B-A3B to produce semantic context of real-world black-box optimization problems, along with 100 evaluated points in the function space. We show two examples of this generated context in Figure 2. By prompting the model to produce example problems with various types of semantic data included, we aim to create training data that matches the diversity of semantic information available in real-world problems. We define the ground truth function space by fitting a surrogate model to these points. By combining GP-generated and LLM-generated objectives, we create a broader and more representative training distribution for fine-tuning.

Trajectory Generation Using our generated function spaces, we generate sequences of proposed configurations and evaluated points, which we call trajectories. We use HEBO (Cowen-Rivers et al., 2022), a state-of-the-art BO framework, to generate training trajectories because we need a high-quality optimizer that includes reliable surrogate modeling capabilities. It has also demonstrated strong empirical performance, including first place in the NeurIPS 2020 Black-Box Optimization Challenge (Turner et al., 2021). We generate our training dataset by running HEBO on our generated function spaces with different surrogate models. We begin by initializing our process with 5 initial steps. We use Sobol sampling (Sobol, 1967) for the GP-based functions and use LLM-generated initializations from Qwen 3.5 35B-A3B for the functions with semantic information. We then proceed with using 3 different surrogate models implemented in HEBO, including Gaussian process (GP), random forest (RF), and Stochastic Variational Inference Deep Kernel Learning (SVIDKL). To compile the dataset, we select the best performing surrogate model after 10, 25, 50, and 100 total steps. Therefore, the model learns from the strongest teacher in each setting.

Prompt Structure An example of the interaction loop is shown in Figure 3. The prompt begins with a system instruction defining the agent's role as a black-box optimization assistant and enforcing a strict JSON output format containing the action and predicted mean and standard deviation. We include a failure probability field after the optimizer has observed at least one failed evaluation. The user then provides the problem description and relevant context. We provide all numbers rounded to 4 significant figures and convert the BO steps into a multi-turn user and assistant conversation. The interaction follows in a turn-based manner where the assistant proposes a new configuration, which is the action proposed by BO and the surrogate model's predicted mean and standard deviation, and the user returns the evaluated objective value, repeating until the evaluation budget is exhausted. Further examples and details of dataset generation are available in Appendix A.

3.2 Training and Inference

Our experiments utilize the Qwen 3.5 27B model (Qwen, 2026). We chose this model for its balance of instruction-following ability and computational accessibility for fine-tuning. Fine-tuning is performed via Low-Rank Adaptation (LoRA) (Hu et al., 2022) leveraging Unsloth (Han et al., 2023). We fine-tune Qwen 3.5 27B on over 100,000 trajectories, with around half of the data containing LLM-generated context. This takes around 2.5 days for the 27B model on 4 Nvidia H200 GPUs.

At inference time, we use our fine-tuned model with the same context structure as training time. We begin our initialization prompt with the relevant context to produce five initialization configura-

```
User: Problem description: Financial Trend Prediction Model Hyperparameter Tuning. Optimize
    the hyperparameters of a LSTM network designed to predict short-term stock market trends.
Tunable search space:
- lstm_units: integer (log search space) in range [16, 256] | description: The number of
    hidden units in the LSTM layer, controlling model capacity. | unit: units
- learning_rate: float (log search space) in range [0.0001, 0.05] | description: The step
    size for the optimizer during model training. | unit: dimensionless
Objective: Direction: maximize | Name: risk_adjusted_return | Unit: ratio | Description: The
    Sharpe Ratio calculated from simulated trades based on the model's trend predictions.
Budget: Total evaluations available (including 5 initial evaluations): 50
Provide 5 initial configurations that explore promising regions in the search space.
Assistant: {"initial_configuration_1": {"lstm_units": 256, "learning_rate": 0.05}, ...,
    "initial_configuration_5": {"lstm_units": 16, "learning_rate": 0.0001}}
User: Initial evaluations:
Evaluation 1: Configuration: {"lstm_units": 256, "learning_rate": 0.05}
Objective (risk_adjusted_return): 0.964 ratio
<Evaluations 2 - 4 omitted>
Evaluation 5: Configuration: {"lstm_units": 16, "learning_rate": 0.0001}
Objective (risk_adjusted_return): 0.45 ratio
Output the next configuration.
Assistant: {"next_configuration": {"lstm_units": 116, "learning_rate": 0.00516}, "mean":
    1.242, "std": 0.0854}
User: Evaluation 6: Objective (risk_adjusted_return): 1.302 ratio
```

Figure 3: An example problem with semantic context from our training data. We fine-tune the model on these text-based sequences and use this structure at inference time to sample new configurations.

tions. We then provide the evaluations for these initial points and iteratively propose and evaluate configurations until the evaluation budget is reached. For each proposal, we sample two possible configurations from the LLM and select the configuration with the higher expected improvement with failure conditioning from the model's predicted distribution, as seen in OptFormer (Chen et al., 2022). More details about the training and inference process are available in Appendix B.1.

4 RESULTS

Baselines and Benchmarks We show the impact of fine-tuning SemanticOpt against our reference comparisons, including the base Qwen 3.5 27B model and the BO methods used to generate our dataset. We utilize implementations from Optuna (Akiba et al., 2019) and OptunaHub (Ozaki et al., 2025) for HEBO (Cowen-Rivers et al., 2022) (with GP, RF, and SVIDKL surrogate models). The goal here is to show that fine-tuning on structured synthetic data gathered from expert numerical methods like BO allows LLMs to gain many of the desirable properties of the numerical optimizer. To aggregate performance across objective scales, we orient all objectives so that higher is better, then normalize between a typical search-space value and a fixed best-known reference:

$$S = (f_{\text{best}} - f_m)/(f_{\text{ref}} - f_m),$$

where f_{best} is the best objective found by the optimizer within the search budget, f_m is the median objective over a fixed 65,536 evaluation Sobol sample and f_{ref} is the best objective found by a 262,144 evaluation evolution-strategy search on the fitted surrogate. This procedure follows standard practice in black-box optimization benchmarking (Turner et al., 2021; Pfisterer et al., 2022). We clip scores below -1 to prevent extreme outliers having an outsized effect.

Semantic Example To illustrate how semantic information can aid optimization, we consider a paint-mixing task in which the optimizer must combine five paints to match a target color without knowing the mixing function. SemanticOpt receives the paint names and descriptions, providing information unavailable to numerical BO. For example, we can provide the color: "Phthalo Green" and a description: "strong vivid green paint" within the prompt to the model. This context allows the model to have more information than just numerical data, allowing the LLM-based approaches to outperform BO methods. We demonstrate this with an illustrative example in Figure 4.

SemanticBBOBench We evaluate SemanticOpt on SEMANTICBBOBENCH (Anonymous, 2026), a benchmark for semantic black-box optimization introduced in concurrent work, which is attached in the supplementary material. The benchmark comprises 42 optimization problems drawn from scientific, engineering, economic, and computational applications, which are completely separate

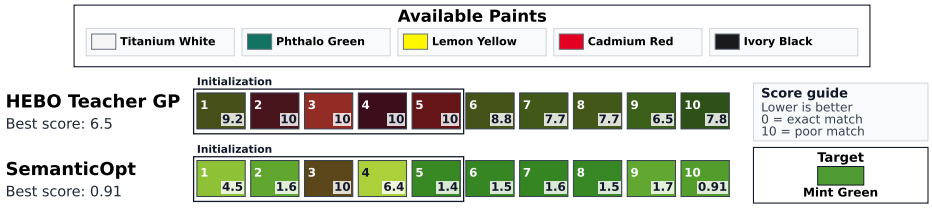


Figure 4: Illustrative example testing finding a mixing ratio of available paints to match the target paint. SemanticOpt shows better initialization and performance given the semantic context than HEBO Teacher GP.

Table 1: Domains represented in SEMANTICBBOBENCH.

Domain	Representative problems
Energy & Infrastructure	Photovoltaics, wind-farm layout and wake steering, networks, charging, geothermal fields, anaerobic digestion, buildings
Thermals & Systems	Electrolyzers, thermoelectrics, power cycles, refrigeration, heat exchangers, heat sinks, desalination
Materials & Life Sciences	Reactions, combustion, nanoalloys, molecular design, metabolic engineering, crop management, measurement
Mechanics & Control	Aircraft missions, airfoils, interplanetary transfers, satellite constellations, vehicle powertrains, trusses, controller tuning
Optics & Acoustics	Antennas, matching networks, lenses, optical coatings, loudspeakers
Operations & Economics	Supply chains, pump scheduling, patient flow, pricing, portfolios, housing design

from the training data for SemanticOpt. The problems span simulator-based and data-derived objectives, contain between 3 and 29 decision variables, and include continuous, integer, categorical, and Boolean parameters, with many tasks also containing infeasible regions. An overview of the benchmark domains is shown in Table 1. Each benchmark provides controlled variants along two semantic axes. The context level axis controls how much problem-specific information is available. *No-context* anonymizes parameter identities and removes the task description while retaining the information required to construct valid configurations. *Base-context* adds meaningful parameter names, units, and descriptions of the task, variables, and objective. Finally, *full-context* provides long-form domain knowledge and descriptions and observations of auxiliary evaluation metrics.

The context type axis tests whether performance depends on the wording or correctness of this information. *Counterfactual* context rewrites the semantic information while preserving the underlying domain relationships and quantitative claims, reducing reliance on surface-form recognition. *Adversarial* context preserves the problem and search space but replaces optimization-relevant guidance with plausible yet misleading information. *Mismatched* context instead provides semantic information from an unrelated problem with a compatible numerical search space. Together, these variants allow us to measure both how SemanticOpt benefits from semantic information and how robustly it revises that information in response to observed evaluations.

Context Level Results Figure 5 shows how performance changes as each model is given more information about the problem. Both models improve steadily with context. The largest step comes from *no-context* to *base-context*, where meaningful parameter names, units, and a task description alone let the model place its early configurations in sensible regions. *Full-context* adds domain knowledge and auxiliary observations, which give a further and smaller gain. Early in the search, the base model is slightly ahead at every context level. One possible cause is that the initial configurations in our training trajectories were generated by Qwen 3.5 35B-A3B due to computational constraints, so fine-tuning partly teaches SemanticOpt to imitate a weaker initializer than the base model. General drift from fine-tuning may also contribute. However, with more evaluations, SemanticOpt overtakes Qwen 3.5 27B and ends ahead at every context level. The advantage is largest

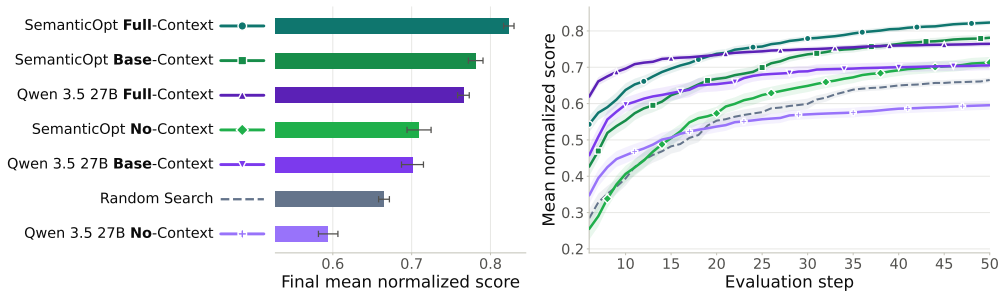


Figure 5: Increasing context level improves performance for both SemanticOpt and Qwen 3.5 27B baselines. After 50 evaluations, SemanticOpt outperforms Qwen 3.5 27B across all context levels. Bars and curves show the mean and standard error over five random seeds per benchmark.

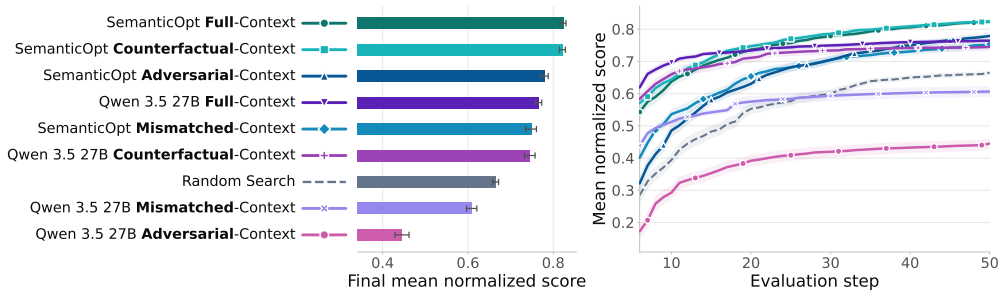


Figure 6: SemanticOpt is more robust than the base model to counterfactual, mismatched, and adversarial semantic information, although misleading context still reduces performance. Curves and bars show the mean and standard error over five random seeds per benchmark.

with *no-context*. There the base model falls below random search, while SemanticOpt remains competitive because it can still learn from the observations it collects.

Context Type Results Figure 6 shows how each model responds when the semantic information is reworded, irrelevant, or wrong. *Counterfactual* rewriting leaves both models essentially unchanged, so neither one depends on recognizing the benchmark's original wording. What matters is the meaning of the context. *Mismatched* and *adversarial* context are where the two models diverge. The base Qwen 3.5 27B model fails to revise its search even after its evaluations contradict it, and it falls below random search in both cases. SemanticOpt is also affected, but it treats the context as a prior to be tested rather than an instruction to be followed. With *adversarial* context it starts well behind its *full-context* counterpart, then closes most of the gap as observations accumulate. *Mismatched* context leaves a more persistent gap. One explanation is that guidance about an unrelated problem results in claims more difficult to refute, while misleading claims about the true problem are exposed within the first few evaluations. Overall, SemanticOpt's advantage over the base model grows as the context becomes less reliable. Fine-tuning keeps the model's ability to use semantic information and teaches it to set that information aside when the numerical evidence disagrees.

Comparison with BO Figure 7 compares the full-context language models with the HEBO teachers, which use only the numerical search space. Both language models start well ahead of the teachers because of their context-informed initializations, and the teachers close most of this gap as observations accumulate. Through 50 evaluations, SemanticOpt continues to improve after initialization while the base Qwen 3.5 27B model plateaus below it.

Disentangling Initialization and Iterative Refinement To separate initialization from iterative refinement, we evaluate each LLM against HEBO continuations from its own five initial configurations. For each model, the LLM and HEBO therefore start from exactly the same five evaluated points, allowing the subsequent performance gap to reflect refinement rather than warm-start quality. Under full and counterfactual context, both LLMs outperform the HEBO teachers during the early

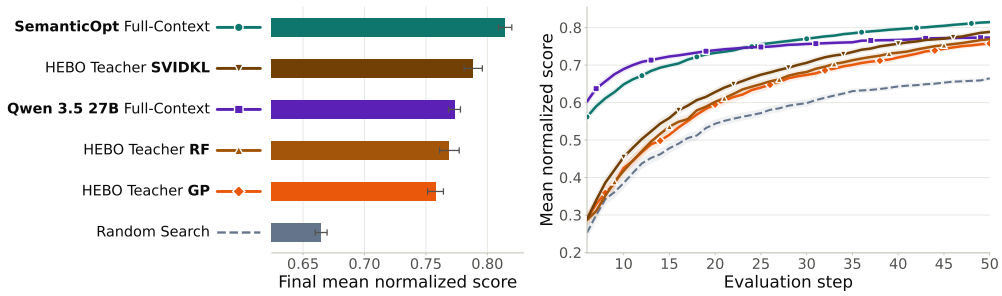


Figure 7: Both language models begin far above the numerical optimizers, and the gap closes as the budget grows. SemanticOpt ends ahead of Qwen 3.5 27B, HEBO Teacher GP and HEBO Teacher RF, and level with HEBO Teacher SVIDKL. Bars and curves show the mean and standard error over ten random seeds per benchmark.

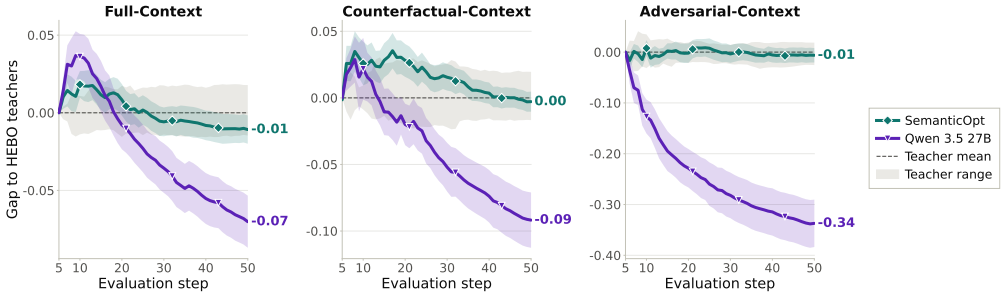
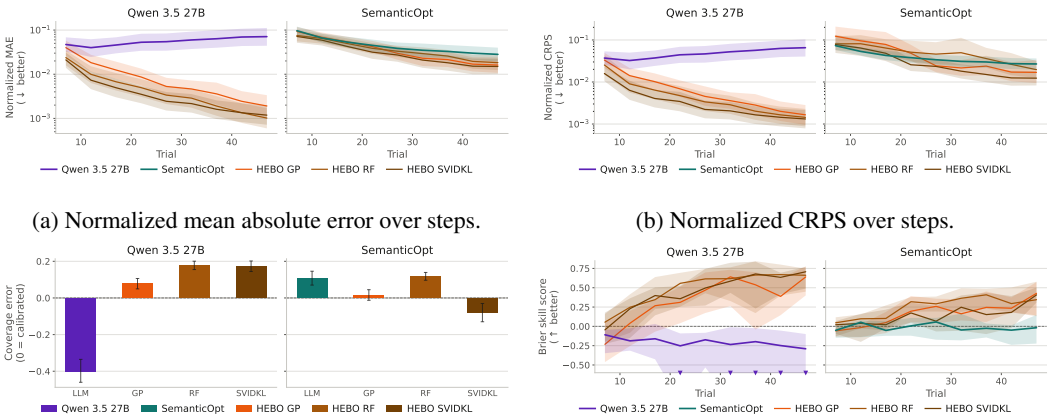


Figure 8: Refinement gap between each LLM and HEBO continuing from that LLM's own five initial configurations. Zero indicates that the LLM refines as well as its teachers. Positive early gaps under informative context show that semantic information continues to help beyond initialization. Base Qwen 3.5 27B increasingly falls behind the teacher methods, whereas SemanticOpt converges to approximately teacher-level refinement across context types. Curves show the mean paired difference over benchmarks with standard-error bands (ten seeds for full context, five otherwise).

post-initialization steps, showing that semantic information continues to help beyond initialization. Under adversarial context, this advantage disappears. As observations accumulate, the base Qwen 3.5 27B model falls behind the BO methods, particularly under adversarial context, while SemanticOpt finishes within approximately 0.01 normalized score of its teachers across all three context types, demonstrating substantially improved iterative refinement after fine-tuning.

Objective Prediction and Uncertainty Figure 9 compares the predictive distribution stated by each LLM against three HEBO surrogates refit on the same history at every trial. The base model and the surrogates respond to accumulating evidence in opposite ways. The base model's prediction error grows over a run, while every surrogate's error falls by more than an order of magnitude (Figure 9a). Fine-tuning reverses this trend, and SemanticOpt's predictions improve alongside the surrogates as observations accumulate. The gap narrows further when the stated uncertainty is scored jointly with the mean. Under CRPS (Figure 9b), SemanticOpt performs similarly to the BO surrogates. In Figure 9c, the base model's intervals are far too narrow, whereas SemanticOpt's are closer to nominal coverage. Failure prediction is the one component where fine-tuning does not close the gap. As shown in Figure 9d, the surrogates surpass the base-rate forecast and keep improving, while the base model degrades. SemanticOpt improves but performs worse than the BO surrogates.

Limitations SemanticOpt improves several aspects of LLM-based optimization, including iterative refinement, predictive accuracy, and uncertainty calibration, but important gaps remain. In particular, its failure-probability predictions remain weaker than those of the HEBO surrogates, suggesting that feasibility modeling is more difficult to acquire through trajectory fine-tuning alone. Fine-tuning also reduces some of the base model's initialization performance, indicating that the current training procedure does not fully preserve its semantic warm-start capability. LLM initial-



(a) Normalized mean absolute error over steps.

(b) Normalized CRPS over steps.

(c) Coverage error of the stated $\pm 1\sigma$ interval.

(d) Failure-probability Brier skill.

Figure 9: Fine-tuning makes the model a better-calibrated in-context predictor. Each panel scores one term of the stated predictive distribution against three HEBO surrogates refit at every trial on the same history: the mean in (a), the mean and uncertainty jointly in (b), the uncertainty alone in (c), and the failure probability in (d). The two LLMs propose different configurations with different distributions, so only comparisons from each LLM to HEBO surrogate methods given the same history are shown. Errors are normalized by each benchmark's feasible objective range, and bands are bootstrap 95% confidence intervals.

ization followed by a numerical optimizer therefore remains a strong alternative, motivating future work on retaining initialization quality while learning stronger iterative refinement. Performance also varies across problem classes. Because the model learns from BO-generated trajectories and LLM-generated synthetic problems, its behavior may depend on the strengths and limitations of the HEBO teachers. We also do not benchmark against other LLM-based optimizers, since our goal is to measure the effect of the training signal rather than to rank optimizers.

Future Work Future work could study broader training distributions, larger base models, alternative numerical teachers, and training objectives that more directly target long-horizon decision making and feasibility estimation. An important next step is to evaluate whether optimization-specific trajectory training transfers to real-world experimental problems, where larger models may better exploit rich domain knowledge while adapting to accumulating numerical evidence. Another promising direction is to study settings that are difficult to express with standard numerical optimizers, including changing search spaces, richer experiment descriptions, and qualitative or partially structured observations. Combining BO-supervised models with inference-time frameworks such as LLAMBO (Liu et al., 2024) or LGBO (Yuan et al., 2026) is another natural direction.

5 Conclusion

We introduced SemanticOpt, an LLM-based black-box optimizer trained on structured optimization trajectories augmented with semantic context and BO surrogate-model predictions. Across 42 scientific and engineering benchmarks, optimization-specific fine-tuning substantially changes how the model uses accumulating numerical evidence. Compared with the base Qwen 3.5 27B model, SemanticOpt trades some initialization quality for substantially stronger iterative refinement, with predictions becoming more accurate and uncertainty estimates better calibrated as observations accumulate. We show that fine-tuning on BO trajectories changes how the model responds to numerical evidence more than it changes what the model knows: iterative refinement, calibration, and robustness to misleading context all improve, while the pretrained model's semantic priors are largely preserved. Numerical optimizers are therefore a practical source of supervision for language-model optimizers, teaching the sequential behavior they lack without displacing the natural-language reasoning they already have. This suggests a path toward optimization methods that combine semantic reasoning with learned numerical adaptation in a single model.

AI USE STATEMENT

In this work, we used generative AI tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, designing and providing feedback on research methodology or experiments, proposing or refining hypotheses, implementing methods, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, and interpreting results. We have not used generative AI tools for assisting with translation or formulating mathematical claims. Providing critical ingredients for proving mathematical claims, and assisting in the writing of proofs are not applicable to this work. Additionally, we used generative AI tools for creating or modifying scientific figures or images, suggesting experimental parameters, creating or editing software code, creation of artifacts, drafting parts of a research paper, summarizing or analysing existing literature, discover research topics or identify gaps, brainstorming, sourcing/searching for information, and identifying relevant literature. We have reviewed all AI-assisted work. The LLM-generated code in this work was reviewed and tested for correctness. AI-tools were used for light editing of this document. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

SemanticOpt and the associated evaluations are created for the purpose of the advancement of understanding methodology capabilities. We do not see any ethical concerns related to this.

REPRODUCIBILITY STATEMENT

The code and data for this paper are available at `https://anonymous.4open.science/r/semanticopt_iclr2027-8176/README.md`. The dataset-generation procedure, including the GP sampling and augmentation scheme, the LLM-generated function spaces, and the surrogate families used to define ground-truth objectives, is described in Appendix A, with the exact composition of the training trajectories in Table 2. Training hyperparameters are listed in Table 3 and the inference-time selection rule in Appendix B.2. The prompt format used at both training and inference time is reproduced in Figures 11 and 12. Every reported comparison is paired over the same seeds. Source code for trajectory generation, fine-tuning, and evaluation is provided as anonymized supplementary material.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for bayesian optimization. *Advances in Neural Information Processing Systems*, 36:20577–20612, 2023.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

Anonymous. Semanticbbobench: A benchmark for semantic black-box optimization with large language models, 2026. Manuscript under review.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020.

Chih-Yu Chang, Milad Azvar, Chinedum Okwudire, and Raed Al Kontar. Llinbo: Trustworthy llm-in-the-loop bayesian optimization. *arXiv preprint arXiv:2505.14756*, 2025.

Sofian Chaybouti, Ludovic Dos Santos, Cedric Malherbe, and Aladin Virmaux. Meta-learning of black-box solvers using deep reinforcement learning. In *NeurIPS 2022, MetaLearn Workshop*, 2022.

Yutian Chen, Xingyou Song, Chansoo Lee, Zi Wang, Richard Zhang, David Dohan, Kazuya Kawakami, Greg Kochanski, Arnaud Doucet, Marc'aurelio Ranzato, et al. Towards learning universal hyperparameter optimizers with transformers. *Advances in Neural Information Processing Systems*, 35:32053–32068, 2022.

Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. In *International Conference on Machine Learning*, pp. 844–853. PMLR, 2017.

Alexander I Cowen-Rivers, Wenlong Lyu, Rasul Tutunov, Zhi Wang, Antoine Grosnit, Ryan Rhys Griffiths, Alexandre Max Maraval, Hao Jianye, Jun Wang, Jan Peters, et al. Hebo: Pushing the limits of sample-efficient hyper-parameter optimisation. *Journal of Artificial Intelligence Research*, 74:1269–1349, 2022.

Fabio Ferreira, Lucca Wobbe, Arjun Krishnakumar, Frank Hutter, and Arber Zela. Can llms beat classical hyperparameter optimization algorithms? a study on autoresearch. *arXiv preprint arXiv:2603.24647*, 2026.

Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and David Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1487–1495, 2017.

Daniel Han, Michael Han, and Unsloth team. Unsloth, 2023. URL `http://github.com/unslothai/unsloth`.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Beichen Huang, Xingyu Wu, Yu Zhou, Jibin Wu, Liang Feng, Ran Cheng, and Kay Chen Tan. Exploring the true potential: Evaluating the black-box optimization capability of large language models. *arXiv preprint arXiv:2404.06290*, 2024.

Carl Hvarfner, Danny Stoll, Artur Souza, Marius Lindauer, Frank Hutter, and Luigi Nardi. pibo: Augmenting acquisition functions with user beliefs for bayesian optimization. *arXiv preprint arXiv:2204.11051*, 2022.

Huaijun Jiang, Yu Shen, Yang Li, Beicheng Xu, Sixian Du, Wentao Zhang, Ce Zhang, and Bin Cui. Openbox: A python toolkit for generalized black-box optimization. *Journal of Machine Learning Research*, 25(120):1–11, 2024.

Agustinus Kristiadi, Felix Strieth-Kalthoff, Marta Skreta, Pascal Poupart, Alán Aspuru-Guzik, and Geoff Pleiss. A sober look at llms for material discovery: Are they actually good for bayesian optimization over molecules? *arXiv preprint arXiv:2402.05015*, 2024.

Robert Lange, Yingtao Tian, and Yujin Tang. Large language models as evolution strategies. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 579–582, 2024.

Ke Li and Jitendra Malik. Learning to optimize. *arXiv preprint arXiv:1606.01885*, 2016.

Xiaobin Li, Kai Wu, Yujian Betterest Li, Xiaoyu Zhang, Handing Wang, and Jing Liu. Pretrained optimization model for zero-shot black box optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. Smac3: A versatile bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23 (54):1–9, 2022. URL `http://jmlr.org/papers/v23/21-0888.html`.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance bayesian optimization. *arXiv preprint arXiv:2402.03921*, 2024.

Alexandre Maraval, Matthieu Zimmer, Antoine Grosnit, and Haitham Bou Ammar. End-to-end meta-bayesian optimisation with transformer neural processes. *Advances in Neural Information Processing Systems*, 36, 2024.

Jamison Meindl, Yunsheng Tian, Tony Cui, Veronika Thost, Zhang-Wei Hong, Johannes Dürholt, Jie Chen, Wojciech Matusik, and Mina Konaković Luković. Zeroshotopt: Towards zero-shot pretrained models for efficient black-box optimization. *arXiv preprint arXiv:2510.03051*, 2025.

Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. Pfns4bo: In-context learning for bayesian optimization. In *International Conference on Machine Learning*, pp. 25444–25470. PMLR, 2023.

Yoshihiko Ozaki, Shuhei Watanabe, and Toshihiko Yanase. OptunaHub: A platform for black-box optimization. *arXiv preprint arXiv:2510.02798*, 2025.

Florian Pfisterer, Lennart Schneider, Julia Moosbauer, Martin Binder, and Bernd Bischl. Yahpo gym – an efficient multi-objective multi-fidelity benchmark for hyperparameter optimization, 2022. URL `https://arxiv.org/abs/2109.03670`.

Qwen. Qwen3.5: Towards native multimodal agents. *URL: https://qwen.ai/blog*, 2026.

Bojana Ranković, Ryan-Rhys Griffiths, and Philippe Schwaller. Large language models as uncertainty-calibrated optimizers for experimental discovery. *Nature Machine Intelligence*, pp. 1–12, 2026.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175, 2015.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

Ilya M Sobol. Distribution of points in a cube and approximate evaluation of integrals. *USSR Computational mathematics and mathematical physics*, 7:86–112, 1967.

Lei Song, Chenxiao Gao, Ke Xue, Chenyang Wu, Dong Li, Jianye Hao, Zongzhang Zhang, and Chao Qian. Reinforced in-context black-box optimization. *arXiv preprint arXiv:2402.17423*, 2024a.

Xingyou Song, Qiuyi Zhang, Chansoo Lee, Emily Fertig, Tzu-Kuo Huang, Lior Belenki, Greg Kochanski, Setareh Ariafar, Srinivas Vasudevan, Sagi Perel, et al. The vizier gaussian process bandit algorithm. *arXiv preprint arXiv:2408.11527*, 2024b.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Yunsheng Tian, Mina Konaković Luković, Timothy Erps, Michael Foshey, and Wojciech Matusik. Autooed: Automated optimal experiment design platform. *arXiv preprint arXiv:2104.05959*, 2021.

Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In *NeurIPS 2020 Competition and Demonstration Track*, pp. 3–26. PMLR, 2021.

Xilu Wang, Yaochu Jin, Sebastian Schmitt, and Markus Olhofer. Recent advances in bayesian optimization. *ACM computing surveys*, 55(13s):1–36, 2023.

Zhuo Yang, Daolang Wang, Lingli Ge, Beilun Wang, Tianfan Fu, and Yuqiang Li. Reasoning bo: Enhancing bayesian optimization with long-context reasoning power of llms. *arXiv preprint arXiv:2505.12833*, 2025.

Xinzhe Yuan, Zhuo Chen, Jianshu Zhang, Huan Xiong, Nanyang Ye, Yuqiang Li, and Qinying Gu. Unleashing llms in bayesian optimization: Preference-guided framework for scientific discovery. *The Fourteenth International Conference on Learning Representations*, 2026.

A Dataset Generation

Function Space Generation Because widely available real-world benchmark functions are limited, we construct a large-scale training dataset using realistic and diverse synthetic objectives. We generate two classes of evaluation functions for this purpose: Gaussian process (GP)-based functions and LLM-generated function spaces. Together, these sources provide both scalable synthetic diversity and semantically grounded problem structures. GPs serve as flexible function generators that represent a wide variety of complex functions, enabling large-scale data collection while producing rich optimization landscapes. The motivation is that GP-based Bayesian optimization methods are most widely used as they perform well across many function classes (Shahriari et al., 2015; Wang et al., 2023). Therefore, by training on GP-generated landscapes, our objective is to learn a policy that can be similarly generalized in diverse problem spaces. To further diversify these tasks, we vary both input and output scales to reflect realistic regimes. For example, objectives may lie in ranges such as $[0, 100]$ or $[1, 10]$, while parameters may span $[10^{-4}, 10^{-1}]$. We augment sampled GP functions with nonlinear coordinate warping, discontinuities, and constraints, transforming smooth samples into landscapes that better resemble real-world systems. We randomly generate a large number of random scales to allow for generalization. Our GP-based functions are sampled from a Gaussian process prior defined over the input space. Each function's covariance kernel is the sum of one or two base kernels sampled from RBF, Matérn 3/2, Matérn 5/2, rational quadratic, exponential, periodic, linear, and polynomial kernels. Hyperparameters such as lengthscales and variances are drawn from log-uniform distributions. A fixed set of initial points is sampled, and function values are generated from the corresponding multivariate Gaussian. Evaluations are obtained by conditioning on this prior, yielding smooth but diverse optimization landscapes.

We also add further augmentations to the randomly generated GP processes. These augmentations transform smooth GP samples into challenging, realistic landscapes by applying non-linear coordinate warping, introducing discontinuities, and adding complex parameter couplings. The goal of this process is to create a training set that mimics the irregularity of real-world systems (e.g., phase transitions, saturation limits, or discrete configuration spaces). By creating functions with these diverse methods, we provide a base for the model to learn realistic function dynamics across a variety of possible function spaces. Lastly, we augment examples with failure regions, where the function space returns failure cases. These are selected as the worst performing points or randomly selected edge regions.

We also use LLM-generated function spaces to introduce real-world domains into the dataset. To do this, we prompt Qwen 3.5 35B-A3B to produce semantic context of real-world black-box optimization problems, along with 100 evaluated points in the function space. We choose this model for its combination of speed and capabilities. By prompting the model to produce example problems with various types of semantic data included, we aim to create training data that matches the diversity of semantic information available in real-world problems.

Depending on the context profile, the generated artifact may also include operational background, a full-paper-style domain brief, objective ranges, historical heuristics, analogous successful parameterizations, and additional diagnostic evaluation metrics. These profiles allow us to construct controlled variants that expose different amounts and types of semantic information, matching the heterogeneous context available in real-world optimization workflows. The sampled points are validated against the declared search space and then used to fit a tabular surrogate objective; the benchmark code selects among candidate surrogate families including XGBoost, CatBoost, Gaussian processes, random forests, and extra trees. The fitted surrogate defines the ground-truth function surface used during optimization, while the generated JSON supplies the semantic context given to context-aware samplers. By combining GP-generated and LLM-generated objectives, we create a broader and more representative training distribution for model fine-tuning.

Trajectory Generation Using our generated function spaces, we generate sequences of proposed configurations and evaluated points, which we call trajectories. We use HEBO (Cowen-Rivers et al., 2022), a state-of-the-art BO framework, to generate training trajectories because we need a high-quality optimizer that includes reliable surrogate modeling capabilities. HEBO is a strong choice because it was designed for practical black-box optimization settings where search spaces are often continuous, discrete, and categorical. It has also demonstrated strong empirical performance, including first place in the NeurIPS 2020 Black-Box Optimization Challenge (Turner et al., 2021).

We generate our training dataset by running HEBO on our generated function spaces with different surrogate models. We begin by initializing our process with 5 initial steps. We use Sobol sampling (Sobol, 1967) for the GP-based functions and use LLM-generated initializations for the functions with semantic information. We then proceed with using 3 different surrogate models implemented in HEBO, including Gaussian process (GP), random forest (RF), and Stochastic Variational Inference Deep Kernel Learning (SVIDKL). To compile the dataset, we select the best performing surrogate model after 10, 25, 50, and 100 total steps. The performance of each surrogate model on our training dataset is shown in Figure 10. These results show that different surrogate models perform better in different function spaces. Additionally, we see that the performance is relatively balanced and the model is learning from a combination of all three surrogate models. This exposes the model to strong and diverse optimization behaviors across function spaces and evaluation horizons.

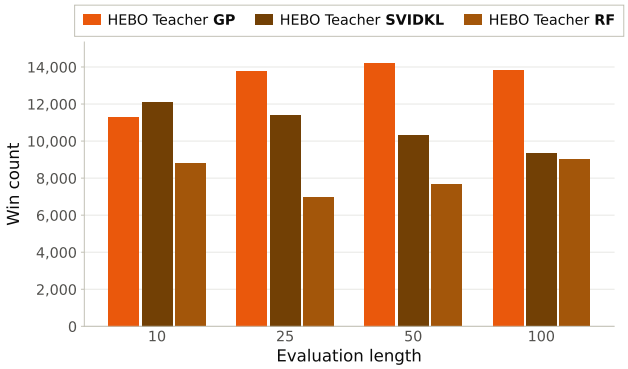


Figure 10: Win count of different HEBO surrogate models. This measures the number of trajectories generated by each HEBO method which are selected as the best trajectory on a function space for each evaluation length. Only the best-performing trajectory for each function space and evaluation length is included in the training data.

We show complete examples of a generated trajectory without context in Figure 11 and a full initialization with context in Figure 12. The prompt begins with a system instruction defining the agent's role as a black-box optimization assistant and enforcing a strict JSON output format containing the action and predicted mean and standard deviation. We include a failure probability field after the optimizer has observed at least one failed evaluation. The user then provides the problem description and relevant context. We provide all numbers rounded to 4 significant figures and convert the BO steps into a multi-turn user and assistant conversation. The interaction follows in a turn-based manner where the assistant proposes a new configuration, which is the action proposed by BO and the surrogate model's predicted mean and standard deviation, and the user returns the evaluated objective value, repeating until the evaluation budget is exhausted.

Table 2 summarizes the trajectory blocks used to construct the training data. For the real-world surrogate data, we include examples of function spaces with and without regions with failure cases, giving the factor of two in the real-world total. These failure regions are returned as unknown constraint failures to allow the model to learn function spaces where failures may occur. Each function space is sampled across nine input dimensions, $d \in \{2, \ldots, 10\}$, with the per-dimension counts shown in the table. The real-world blocks therefore contribute $2 \cdot 9 \cdot 800 = 14{,}400$ function spaces. The synthetic data uses four dataset families, with 500 functions per dimension, contributing $4 \cdot 9 \cdot 500 = 18{,}000$ function spaces. In total, this yields $32{,}400$ function spaces. Since we select the best trajectory in the function space from four trajectory lengths, $T \in \{10, 25, 50, 100\}$, the final dataset contains $32{,}400 \cdot 4 = 129{,}600$ training sequences.

B Model Training and Usage

B.1 Training

We use the Qwen 3.5 family of models as our base model. Specifically, we fine-tune the Qwen 3.5 27B model, which provides strong instruction-following and numerical reasoning capabilities while

```
System:
You are a black-box optimization assistant. Your goal is to iteratively propose
    configurations to optimize the objective within an evaluation budget.
At each step, return exactly one valid JSON object as your output.
 - Before any evaluations are observed, if the user asks for initialization, the JSON must
    contain exactly these keys: initial_configuration_1, ..., initial_configuration_5.
 - Before any observed failure, the JSON must contain exactly these keys: next_configuration,
    mean, std.
 - After at least one observed failure, the JSON must contain exactly these keys:
    next_configuration, mean, std, failure_probability.
 - next_configuration must include every key in the tunable search space with valid in-range
    values.
 - mean and std must predict the objective value distribution for the proposed configuration.
Each new configuration should seek to gain new insight about the problem space and achieve
    the best objective possible within the evaluation budget.
User:
Problem description:
We are optimizing an unknown black-box problem.
Tunable search space:
- param_0: float (log search space) in range [0.002758, 0.1874] | description: unknown float
    (log search space) parameter | unit: unknown units
- param_1: float in range [0.4848, 3.843] | description: unknown float parameter | unit:
    unknown units
- param_2: integer in range [3, 24] | description: unknown integer parameter | unit: unknown
    units
- param_3: integer in range [40, 291] | description: unknown integer parameter | unit:
    unknown units
Objective:
- Direction: minimize
- Unit: unknown units
- Description: unknown black-box scalar objective
Budget:
- Total evaluations available (including 5 initial evaluations): 100

Initial evaluations:
Evaluation 1:Configuration: {"param_0": 0.168, "param_1": 2.512, "param_2": 19, "param_3":
    227}
Objective: -0.3808 unknown units
Evaluation 2: Configuration: {"param_0": 0.1219, "param_1": 2.662, "param_2": 18, "param_3":
    204}
Objective: -0.3239 unknown units
Evaluation 3: Configuration: {"param_0": 0.1798, "param_1": 2.53, "param_2": 5, "param_3":
    122}
Objective: failure
Evaluation 4: Configuration: {"param_0": 0.1851, "param_1": 1.963, "param_2": 3, "param_3":
    62}
Objective: -0.1428 unknown units
Evaluation 5: Configuration: {"param_0": 0.1767, "param_1": 1.251, "param_2": 3, "param_3":
    189}
Objective: -0.2731 unknown units

Output the next configuration, including failure_probability.
Assistant:
{"next_configuration": {"param_0": 0.09889, "param_1": 2.057, "param_2": 20, "param_3": 225},
    "mean": -0.1429, "std": 0.2828, "failure_probability": 0.178}
User:
Evaluation 6: Objective: -0.6286 unknown units
Output the next configuration, including failure_probability.
Assistant:
{"next_configuration": {"param_0": 0.1022, "param_1": 1.872, "param_2": 23, "param_3": 260},
    "mean": -0.5211, "std": 0.2647, "failure_probability": 0.1403}
User:
Evaluation 7: Objective: -0.5112 unknown units
Output the next configuration, including failure_probability.
```

Figure 11: A synthetic example problem from our training data. This example contains failure cases, such that the model outputs failure probability for proposed points.

remaining feasible to adapt with parameter-efficient fine-tuning. We fine-tune the model using low-rank adaptation (LoRA) (Hu et al., 2022) with Unsloth (Han et al., 2023). This provides an efficient framework for adapting large language models within our computational budget. We use our generated dataset of conversational optimization trajectories, where each example contains the problem context, previous evaluations, surrogate-model predictions, and the next configuration selected by the optimizer to train the model. We use the resulting model for all SemanticOpt experiments.

```
User:
Problem description:
You are optimizing a black-box benchmark: Online Grocery Delivery Window Optimization.
This task involves tuning operational parameters to minimize customer wait times while
    maximizing driver utilization for same-day grocery deliveries. The system must balance
    aggressive delivery promises against the risk of missing windows due to traffic and
    order volume fluctuations. Key levers include time-slot granularity, buffer allocations,
    and dynamic pricing thresholds that influence demand distribution. The goal is to find a
    configuration that reduces average lateness without causing excessive driver idle time.
In the online grocery sector, delivery window commitments are a critical competitive
    differentiator. Customers expect precise time slots, but operational reality is driven
    by unpredictable variables like traffic, weather, and order clustering. Operators must
    configure their dispatch algorithms to accept orders within specific constraints that
    ensure on-time delivery while maintaining cost efficiency. This optimization problem
    focuses on the interplay between slot definition, buffer management, pricing strategies,
    and route capacity limits. Poorly tuned parameters can lead to a cascade of missed
    deliveries, customer churn, and increased operational costs due to inefficient routing
    or overtime pay.
Tunable search space:
- max_orders_per_route: integer in range [10, 50] | description: Maximum number of orders
    assigned to a single driver route. | unit: orders
- enable_dynamic_pricing: boolean in set {true, false} | description: Whether to activate
    surge pricing for peak demand windows. | unit: unitless
- route_reoptimization_interval: category in set {hourly, every_30min, none} | description:
    How often the routing plan is recalculated during the day. | unit: frequency
- pricing_elasticity: float (log search space) in range [0.2, 1.5] | description: Sensitivity
    of demand to price changes; higher values shift demand to off-peak times. | unit: ratio
- cancellation_threshold_minutes: integer in range [15, 120] | description: Time before the
    window start when an order is automatically cancelled if not confirmed. | unit: minutes
- buffer_minutes: integer in range [0, 30] | description: Extra time added to the promised
    window to absorb minor delays. | unit: minutes
- slot_duration_minutes: integer in range [30, 120] | description: The duration of each
    available delivery time slot. | unit: minutes
- priority_weight_factor: float in range [0.5, 3] | description: Weight applied to VIP orders
    in the routing algorithm. | unit: multiplier
Objective:
- Direction: minimize
- Name: weighted_service_cost
- Unit: cost_units
- Description: A composite metric combining the total minutes of lateness weighted by order
    value and the cost of driver idle time. Lower values indicate better service reliability
    and operational efficiency.
Budget:
- Total evaluations available (including 5 initial evaluations): 100

Provide 5 initial configurations that explore promising regions in the search space.
```

Figure 12: An example initialization with contextual information from our training data.

We use the hyperparameters shown in Table 3 to fine-tune the model. The trajectories are formatted as multi-turn text conversations, as shown in Figure 11, and the model is trained with supervised next-token prediction on the assistant responses only. Training is performed with BF16 precision using LoRA adapters, which enables fine-tuning of the 27B-parameter base model within our available GPU memory. Further hyperparameter tuning and longer training runs were limited by computational budget. We found that higher learning rates improved training loss but resulted in worse initialization performance. Therefore, we achieved a good balance by using a lower learning rate.

B.2 Inference

At inference time, SemanticOpt uses the same multi-turn structure as in training. For problems with semantic context, the model first receives the problem description and relevant contextual information, then proposes 5 initial configurations. After these initial evaluations are returned, SemanticOpt iteratively proposes new configurations until the evaluation budget is reached.

For each optimization step, we sample two candidate configurations from the fine-tuned model. Each candidate includes a proposed configuration, the model's predicted objective mean and standard deviation, and a predicted probability that the evaluation will fail. We then compute expected improvement using the self-predicted objective distribution and weight it by the predicted success probability:

$$\mathbf{x}_{t+1} = \mathbf{x}^{(j^*)}, \qquad j^* = \arg\max_j p_{\text{succ}}^{(j)} \, \text{EI}(\mu^{(j)}, \sigma^{(j)}, y_t^*).$$

Table 2: Function space variants used for training-data generation. All functions are generated for dimensions $d \in \{2, \ldots, 10\}$ and trajectories are selected for trajectory lengths $T \in \{10, 25, 50, 100\}$.

Dataset Family	Function Space Variants	Functions / Dimension
Real-world surrogate	Base context	200
	History	200
	Long-form context	100
	Extra metrics	50
	History + long-form	100
	History + extra metrics	50
	Long-form + extra metrics	50
	All context	50
Synthetic	Base	500
	Augmented	500
	Contains Failure Regions	500
	Augmented + Contains Failure Regions	500
	Real-world Function Spaces	$2 \cdot 9 \cdot 800 = 14{,}400$
	Synthetic Function Spaces	$4 \cdot 9 \cdot 500 = 18{,}000$
	Total Function Spaces	$\mathbf{32{,}400}$
	Total Training Trajectories	$\mathbf{129{,}600}$

Table 3: Hyperparameters for model training.

Hyperparameter	Value
Base Model	Qwen 3.5 27B
Fine-tuning Method	LoRA
Learning Rate	1×10^{-5} with cosine scaling
Batch Size	64 trajectories
LoRA Rank	16
LoRA Alpha	16
Precision	BF16
Epochs	1
Maximum Sequence Length	24,576 tokens

Equivalently, this downweights candidates that may have high predicted objective improvement but are likely to fail during evaluation. This provides a lightweight exploration-exploitation mechanism that also accounts for feasibility, while keeping inference simple and fully contained within the LLM-based optimizer.

Temperature and Candidate Count SemanticOpt has two inference settings that are not learned during fine-tuning: the sampling temperature and the number of candidates k sampled at each optimization step. When $k > 1$, we select among the sampled candidates using the success-weighted expected-improvement rule. We use the same inference settings for SemanticOpt and the base Qwen 3.5 27B model in our main experiments so that their comparison isolates the effect of fine-tuning rather than differences in decoding or candidate selection.

We use a sampling temperature of 1.25 to encourage proposal diversity. At temperature 1.0, we observed repeated or near-duplicate configurations, particularly from the base Qwen model. Increasing the temperature to 1.25 reduced this behavior for both models without causing formatting failures. We additionally evaluate sampling $k = 2$ candidates and selecting between them using the model's predicted objective distribution, similar to the acquisition-based selection used in OptFormer (Chen et al., 2022). This provides a lightweight mechanism for using the model's learned mean and uncertainty estimates during inference without fitting an external surrogate.

Table 4 examines the effect of candidate count during inference. At a fixed temperature of 1.25, selecting between two candidates using success-weighted expected improvement improves SemanticOpt from 0.813 to 0.823 at 50 evaluations, while providing no improvement for the base Qwen

model. This is consistent with Figure 9, where fine-tuning substantially improves the reliability of SemanticOpt's objective and uncertainty predictions. We therefore use $k = 2$ as the default SemanticOpt inference procedure. For controlled comparisons, the base Qwen model is evaluated using the same setting unless otherwise noted. All remaining decoding parameters are fixed at temperature 1.25, top-$p = 0.95$, top-$k = 20$, and presence penalty 1.5. Every configuration in Table 4 is evaluated on all 42 benchmarks with five matched seeds.

Table 4: Sensitivity of SemanticOpt and Qwen 3.5 27B to candidate count k at a fixed sampling temperature of 1.25. When $k = 2$, the candidate with the highest success-weighted expected improvement is selected.

	SemanticOpt		Qwen 3.5 27B	
Candidates	10 evals	50 evals	10 evals	50 evals
$k = 1$	0.620	0.813	0.709	0.778
$k = 2$	0.638	0.823	0.696	0.764

C Baseline Details

All baselines are run on the same benchmarks, evaluation budget, and matched seeds as SemanticOpt. We detail the baseline implementations below.

HEBO Teachers The HEBO teachers use the OptunaHub HEBO sampler and surrogate settings that generated the fine-tuning trajectories, so SemanticOpt is compared against the procedures it was trained to imitate. They see only the numerical search space. Each run begins with five scrambled Sobol points. HEBO then applies a power transform to the observed objectives, fits the surrogate, and optimizes the MACE acquisition (Cowen-Rivers et al., 2022) with NSGA-II, drawing the next configuration from the Pareto front over the lower confidence bound, expected improvement, and probability of improvement. The GP surrogate uses a Matérn-3/2 kernel with automatic relevance determination. The RF surrogate is a random forest whose predictive variance is the variance across trees. SVIDKL places a sparse variational GP on features learned by a small neural network. All surrogate settings are HEBO defaults. Failed evaluations are excluded from the objective surrogate. Once both successes and failures have been observed, a second model of the same surrogate family is regressed on the binary outcomes, and the acquisition becomes expected improvement weighted by the predicted probability of success, the same criterion SemanticOpt applies to its own predictions.

Random Search Random search draws every configuration from a scrambled Sobol sequence over the full search space. It uses no model and ignores failures.

Qwen 3.5 27B The base-model baseline is the untuned Qwen 3.5 27B model that SemanticOpt was fine-tuned from. It receives identical prompts at every context level, uses the same decoding settings, and selects between two sampled candidates by success-weighted expected improvement computed from its own stated predictions. The comparison therefore isolates the effect of fine-tuning.

D Acquisition Gap and a Budgeted Safeguard

One advantage of Bayesian optimization is that its acquisition rule provides a well-defined mechanism for balancing exploration and exploitation, with regret guarantees available for methods such as Gaussian process upper confidence bound (GP-UCB) (Srinivas et al., 2009; Chowdhury & Gopalan, 2017). SemanticOpt instead proposes configurations directly with an LLM, without explicitly maximizing an acquisition function, so these guarantees do not apply directly. In this section, we first measure the *acquisition gap*: how closely SemanticOpt's proposals align with those selected by GP-UCB. We then introduce a lightweight budgeted safeguard that could be added to SemanticOpt, which occasionally defers to GP-UCB, allowing SemanticOpt to retain its semantic proposal mechanism while bounding its regret relative to the GP-UCB reference policy.

Table 5: Proposals scored by a fixed reference GP, averaged over the 42 benchmarks from the tenth evaluation onward. Quantiles are ranks within a shared Sobol pool. Top 5% is the share of proposals whose UCB value is in the top 5% of the pool. Near repeats are proposals within a root-mean-square distance of 0.02 of an earlier evaluation in the unit-scaled search space. Budget $\kappa^\star$ is the median smallest budget at which the safeguard leaves a run unchanged.

Method	UCB quantile	Top 5%	Std. quantile	Near repeats	Budget $\kappa^\star$
SemanticOpt	0.69	33%	0.12	11%	4.8
Qwen 3.5 27B	0.55	28%	0.01	74%	6.8
HEBO Teacher GP	0.68	31%	0.16	25%	4.3
HEBO Teacher RF	0.73	34%	0.16	9%	4.4
HEBO Teacher SVIDKL	0.78	43%	0.20	9%	3.4
Random search	0.53	5%	0.61	0%	7.5

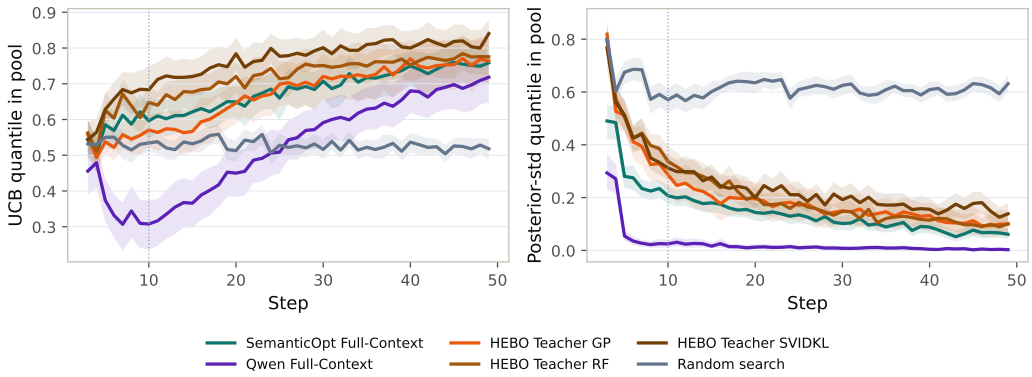


Figure 13: Rank of each proposal within a shared Sobol pool under a reference GP fitted to the history the optimizer had seen. Left: UCB value. Right: posterior standard deviation. Curves show the mean over the 42 benchmarks, and bands show bootstrap 95% intervals.

Measuring the Acquisition Gap We replay every recorded trajectory with a fixed reference GP. At each step, we fit the reference to the successful evaluations observed so far and score the next proposal. The reference is the default BoTorch GP (Balandat et al., 2020) with a dimension-scaled lengthscale prior, one-hot encoded categoricals, and the acquisition $U_t(\mathbf{x}) = \mu_{t-1}(\mathbf{x}) + 2\sigma_{t-1}(\mathbf{x})$. We report where each proposal's UCB value and posterior standard deviation rank within a Sobol pool of 2048 configurations shared by all methods on a benchmark.

Acquisition Gap Results As shown in Table 5 and Figure 13, random search ranks in the middle of the pool and the HEBO teachers rank highest, including the RF and SVIDKL teachers, which use different surrogates. SemanticOpt ranks at or above HEBO Teacher GP throughout the run, but below the RF and SVIDKL teachers. The base Qwen 3.5 27B model instead proposes points where the reference is already certain, ranks below random search for the first half of the run, and nearly repeats an earlier evaluation in about three quarters of its proposals. Fine-tuning removes this behavior. Under a Matérn-5/2 reference kernel, the ordering of SemanticOpt, the base model, and random search is unchanged, although SemanticOpt ranks somewhat below HEBO Teacher GP.

Budgeted Safeguard SemanticOpt samples two candidates per step and selects one by success-weighted expected improvement (Appendix B.2). We add one test at this step. Let $\hat{\mathbf{x}}_t$ be the maximizer of the reference acquisition returned by an acquisition optimizer, and let $s_t = \max\{0, U_t(\hat{\mathbf{x}}_t) - U_t(\mathbf{x}^{(j^*)})\}$ be the shortfall of the selected candidate. The candidate is evaluated if the cumulative shortfall $S_t = S_{t-1} + s_t$ stays within $\kappa\sqrt{t}$; otherwise $\hat{\mathbf{x}}_t$ is evaluated and S_t is unchanged. Setting $\kappa = \infty$ recovers SemanticOpt, and $\kappa = 0$ reduces to GP-UCB. Because the GP-UCB regret analysis only picks up the shortfall of each selected point, under the assumptions of Chowdhury & Gopalan (2017) the safeguarded variant satisfies, with high probability and for any

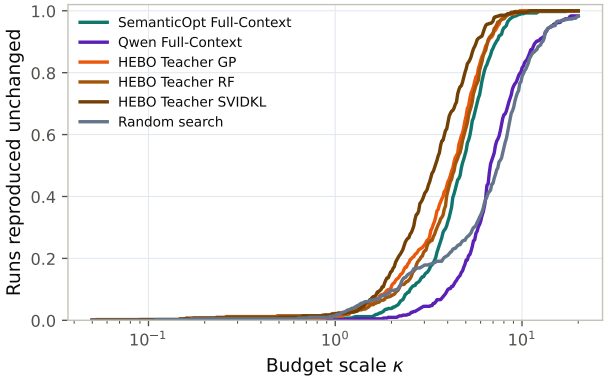


Figure 14: Share of recorded runs left unchanged by the budgeted safeguard as a function of the budget scale κ. A run is unchanged when its cumulative acquisition shortfall stays within $\kappa\sqrt{t}$ at every step. Runs on which the reference GP could never be fitted are excluded.

candidates the language model proposes,

$$R_T \leq \bar{R}_T + \kappa\sqrt{T} + \sum_{t=1}^{T} \eta_t,$$

where $\bar{R}_T$ is the regret bound of GP-UCB and η_t is the acquisition optimizer's error, which GP-UCB also incurs. Since $\bar{R}_T$ grows at least as fast as $\sqrt{T}$, the safeguard keeps the regret rate of GP-UCB even under misleading context (Figure 6). The guarantee is a worst-case floor relative to GP-UCB, not a claim of improvement, and it covers the objective but not failed evaluations. Like πBO (Hvarfner et al., 2022), which trusts a user prior with a decaying weight, it limits how far a prior can pull the search; here the prior is the language model and the limit is a budget on acquisition shortfall.

Budget Needed by the Recorded Runs When the test never fails, the safeguarded variant makes the same choices as SemanticOpt. For each recorded run, we compute $\kappa^\star = \max_t S_t/\sqrt{t}$, the smallest budget that leaves the run unchanged, with shortfalls on the reference GP's standardized outcome scale and the five initial configurations exempt. As shown in Figure 14, SemanticOpt needs slightly more budget than the HEBO teachers, while the base model needs about as much as random search. A budget that leaves more than four fifths of teacher runs unchanged also leaves three quarters of SemanticOpt runs unchanged, against about a third of base-model runs. Because the replay uses $\beta^{1/2} = 2$ rather than the theoretical β_t, $\kappa^\star$ calibrates the budget rather than certifying the bound, and performance at smaller budgets requires running the variant.

A practical barrier to deploying language model optimizers on expensive experiments is that their proposals come with no guarantee, so a misleading description or an unfamiliar problem can waste an entire budget. The budgeted safeguard addresses this barrier. While our main results focus solely on the implementation and performance of SemanticOpt, these examples show extending SemanticOpt to be bounded relative to GP-UCB is possible and would require limited changes to SemanticOpt decisions. Semantic knowledge can steer the search freely where it helps most, early in a run, while the method keeps the worst-case guarantee of the numerical optimizer it falls back on. The budget κ gives practitioners a single, interpretable setting for how much to trust the model, and the recorded runs suggest that a fine-tuned model can be granted a generous budget while an untuned model cannot. The same shortfall measurement applies to any language model optimizer, which makes it a simple test of whether a model has learned to search like a Bayesian optimizer. Natural extensions include constraint-aware acquisitions, so that the guarantee also covers failed evaluations, and budgets that adapt to how often the model's proposals are confirmed by the observations.

E FURTHER RESULTS

Per Benchmark Results The aggregate curves average over benchmarks and can hide whether a method wins broadly or on a few problems only. Figure 15 therefore reports the final score on every benchmark separately, ordered by the score of SemanticOpt with *full-context*. The ordering makes the spread visible: the LLM variants lead on most benchmarks, the margin over HEBO GP is small on the problems where every method saturates near the observed optimum, and the benchmarks where the LLM variants trail are concentrated at the bottom of the ranking, where all methods score poorly.

Per-Domain Results To check whether the gains are driven by a single application area, we group the benchmarks into the six domains of Table 1 (Figure 16). Both language models start ahead of the HEBO teachers in every domain, with the largest early margins in Optics & Acoustics and Materials & Life Sciences. By 50 evaluations the teachers match SemanticOpt in Thermals & Systems and Operations & Economics, while SemanticOpt remains ahead in Energy & Infrastructure, Materials & Life Sciences, and Optics & Acoustics. SemanticOpt ends above the base model in every domain except Optics & Acoustics, where Qwen 3.5 27B's strong initialization persists.

We further compare SemanticOpt benchmark by benchmark rather than in aggregate, taking the full-context arm as the reference and differencing every other arm against it on all 42 benchmarks, at 10 and 50 evaluations. The comparisons fall into three groups. The first varies what SemanticOpt itself is told: two levels below the full context, from *base-context* down to *no-context*, and three context types. The *counterfactual* rewrite removes surface identifying detail while leaving the problem intact, the *mismatched* rewrite replaces the context with the context from a new problem with the same search space format, and the *adversarial* rewrite replaces it with misleading guidance.

The second group repeats all six conditions on Qwen 3.5 27B, the model SemanticOpt was fine-tuned from, which separates what the fine-tuning contributes from what the base model already did. The third is the classical baselines: the three HEBO teacher surrogates whose rollouts the fine-tuning was distilled from.

Every method sees the same benchmarks under the same five seeds, so the comparisons are paired. For each configuration we report the mean difference in normalized score, a percentile bootstrap 95% confidence interval over the 42 benchmark-level differences, and the number of benchmarks on which the difference is positive. Positive values favor SemanticOpt with full context.

Refinement from a Shared Initialization For each benchmark and seed, we replay the first five evaluations of a recorded LLM trajectory into each HEBO teacher (GP, RF and SVIDKL), which then spends the remaining budget. The replayed objective values are checked against the recorded ones, so every teacher continues from exactly the history the LLM saw, and teacher seeds match the standalone teacher runs. Counterfactual and adversarial runs use the reworded search space of the counterfactual task for both the LLM and the teachers. The refinement gap in Figure 8 is the LLM's normalized score minus the mean of the three teacher continuations, averaged over benchmarks, with the standard error of that paired difference. Full context uses the ten-seed runs and the other contexts the five-seed runs due to computational constraints.

Table 8 tests each gap with a paired Wilcoxon signed-rank test over the 42 benchmarks. After 50 evaluations SemanticOpt shows no significant difference from its teachers under any of the three contexts and is ahead on roughly half the benchmarks, while Qwen 3.5 27B trails significantly under all three. After 10 evaluations, both models lead the teachers under full context and trend ahead under counterfactual context, while under adversarial context the base model already trails and SemanticOpt runs level with its teachers, which places the early lead in the semantic information rather than in the models' search behavior.

Absolute trajectories Figure 17 shows the trajectories behind the gaps. SemanticOpt stays within the range of its teachers under each context, ending level with HEBO Teacher GP and RF and slightly behind HEBO Teacher SVIDKL. Under adversarial context, SemanticOpt's own initial proposals are also less damaged than the base model's, and HEBO recovers from either model's misleading start.

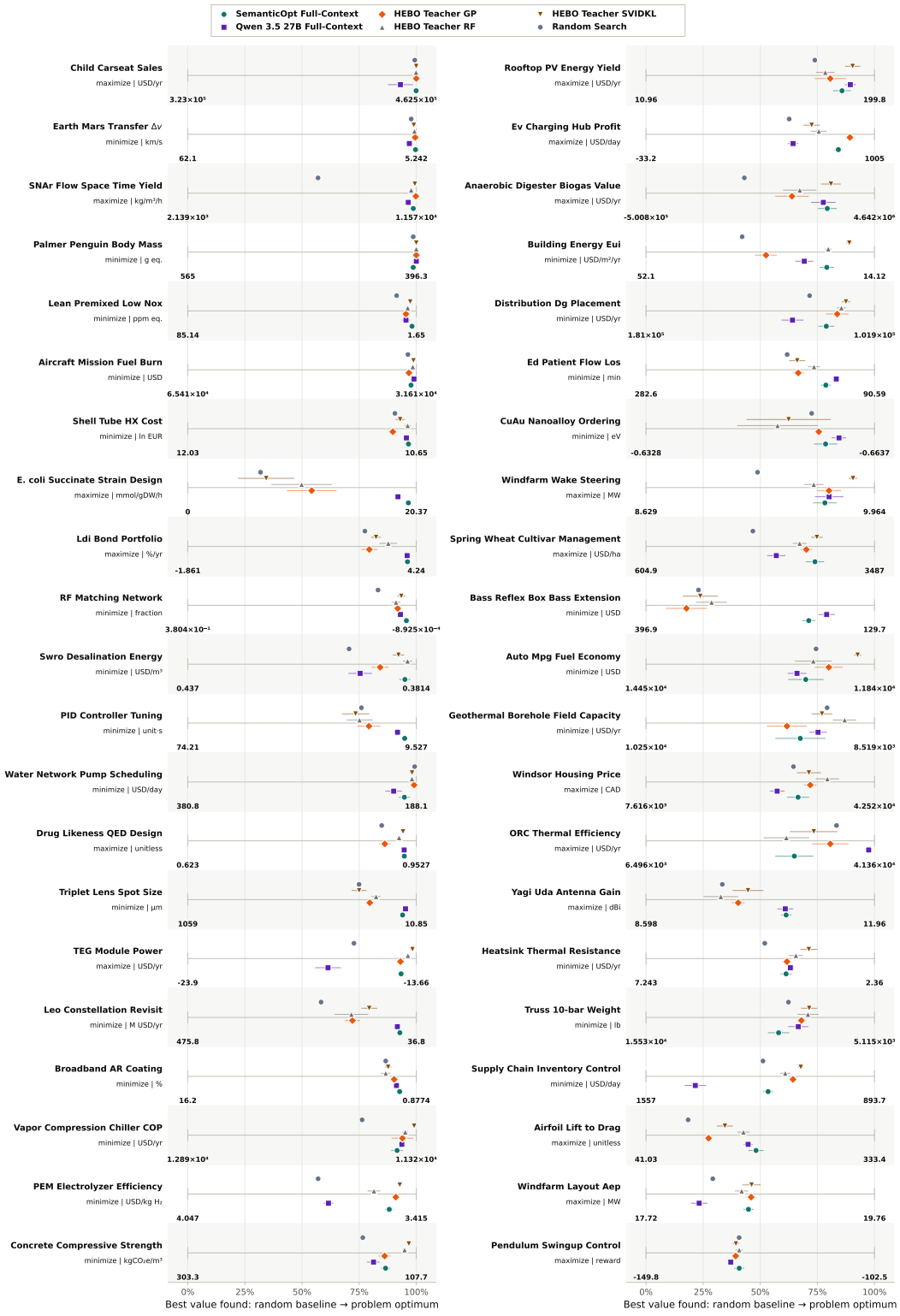


Figure 15: Scores after 50 evaluations on each of the 42 benchmarks, best at the top of the left column and continuing into the right. Markers are the mean over five seeds and bars are one standard error.

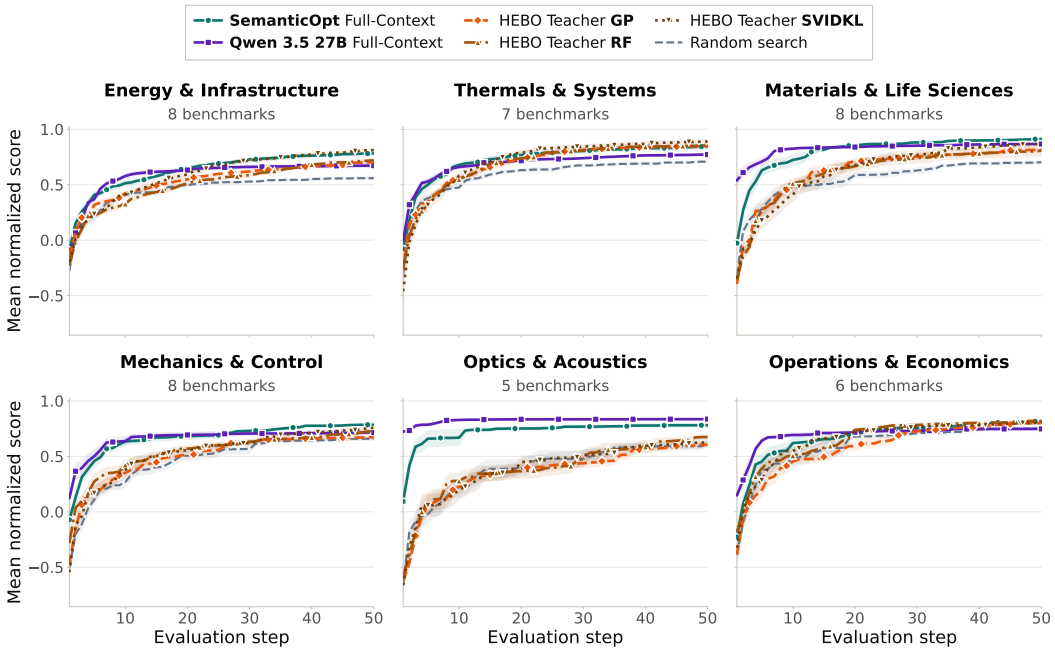


Figure 16: Mean normalized score as a function of evaluation budget within each application domain for SemanticOpt, Qwen 3.5 27B, and the three teacher BO methods. Bands are one standard error over the benchmarks of the domain.

Table 6: Per-benchmark differences relative to SemanticOpt with full context. Each of the 42 benchmarks contributes one value, obtained by averaging its five matched seeds before differencing. Intervals are percentile bootstrap 95% confidence intervals over the 42 benchmark-level differences. Positive values favor SemanticOpt with full context.

	10 evaluations	50 evaluations
Compared arm	Mean Δ [95% CI]	Mean Δ [95% CI]
SemanticOpt		
Base	$+0.084$ $[+0.027, +0.141]$	$+0.042$ $[+0.017, +0.069]$
No-Context	$+0.230$ $[+0.152, +0.317]$	$+0.114$ $[+0.066, +0.167]$
Counterfactual	-0.006 $[-0.052, +0.043]$	$+0.001$ $[-0.018, +0.020]$
Mismatched	$+0.106$ $[+0.049, +0.171]$	$+0.075$ $[+0.034, +0.123]$
Adversarial	$+0.152$ $[+0.089, +0.218]$	$+0.044$ $[+0.021, +0.068]$
Qwen 3.5 27B		
Full	-0.059 $[-0.108, -0.011]$	$+0.059$ $[+0.020, +0.095]$
Base	$+0.042$ $[-0.013, +0.100]$	$+0.122$ $[+0.079, +0.166]$
No-Context	$+0.180$ $[+0.110, +0.254]$	$+0.229$ $[+0.175, +0.290]$
Counterfactual	-0.023 $[-0.069, +0.023]$	$+0.078$ $[+0.034, +0.121]$
Mismatched	$+0.116$ $[+0.045, +0.194]$	$+0.214$ $[+0.152, +0.291]$
Adversarial	$+0.344$ $[+0.249, +0.445]$	$+0.378$ $[+0.286, +0.477]$
BO Teacher baselines		
HEBO Teacher GP	$+0.214$ $[+0.128, +0.309]$	$+0.079$ $[+0.033, +0.129]$
HEBO Teacher RF	$+0.200$ $[+0.123, +0.288]$	$+0.053$ $[+0.019, +0.091]$
HEBO Teacher SVIDKL	$+0.210$ $[+0.121, +0.307]$	$+0.021$ $[-0.015, +0.061]$

LLM initialization followed by HEBO These runs also show that handing an LLM's initial proposals to a numerical optimizer is a strong combination in its own right. HEBO recovers even from the base model's misleading adversarial proposals, and with useful context this hybrid ends at least

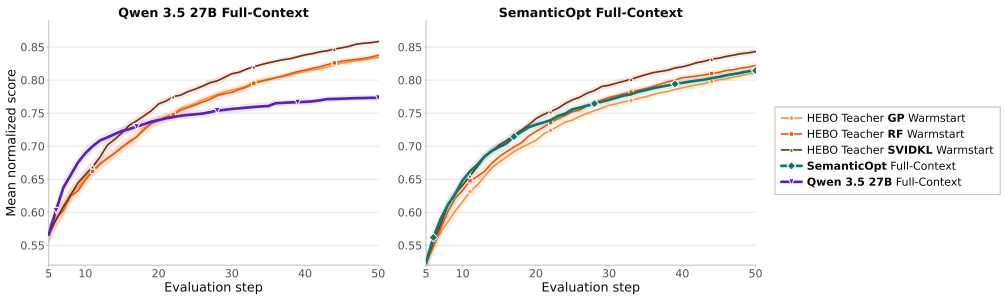
Table 7: Number of benchmarks, out of 42, on which SemanticOpt with full context outperforms each compared arm, based on paired per-benchmark differences.

	Wins out of 42 benchmarks	
Compared arm	10 evaluations	50 evaluations
SemanticOpt		
Base	$30/42$	$27/42$
No-Context	$35/42$	$33/42$
Counterfactual	$22/42$	$23/42$
Mismatched	$30/42$	$29/42$
Adversarial	$34/42$	$33/42$
Qwen 3.5 27B		
Full	$13/42$	$31/42$
Base	$22/42$	$35/42$
No-Context	$32/42$	$40/42$
Counterfactual	$16/42$	$30/42$
Mismatched	$31/42$	$35/42$
Adversarial	$36/42$	$37/42$
BO Teacher baselines		
HEBO Teacher GP	$35/42$	$28/42$
HEBO Teacher RF	$35/42$	$28/42$
HEBO Teacher SVIDKL	$29/42$	$22/42$

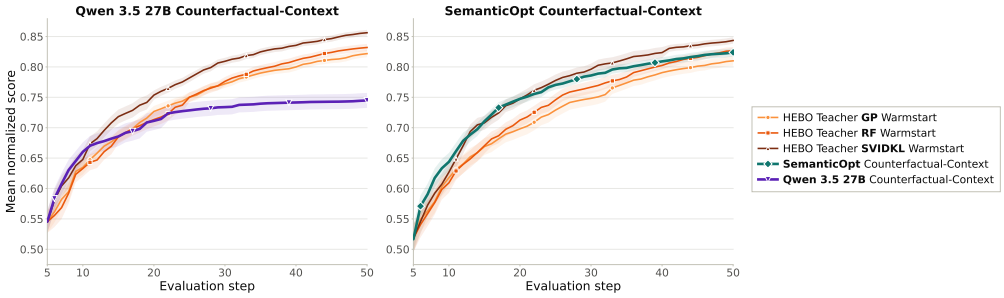
Table 8: Refinement gap to the mean of the three HEBO teachers continuing from the same history, with the number of benchmarks on which the LLM is ahead and the paired Wilcoxon p-value over 42 benchmarks.

		After 10 evaluations			After 50 evaluations		
Context	Model	Gap	Ahead	p	Gap	Ahead	p
Full	SemanticOpt	$+0.018$	35/42	0.0008	-0.011	23/42	0.57
	Qwen 3.5 27B	$+0.036$	27/42	0.016	-0.070	10/42	7×10^{-5}
Counterfactual	SemanticOpt	$+0.026$	27/42	0.084	-0.003	22/42	0.99
	Qwen 3.5 27B	$+0.022$	29/42	0.053	-0.092	6/42	3×10^{-5}
Adversarial	SemanticOpt	$+0.008$	25/42	0.29	-0.006	16/42	0.43
	Qwen 3.5 27B	-0.127	12/42	0.0004	-0.337	5/42	3×10^{-10}

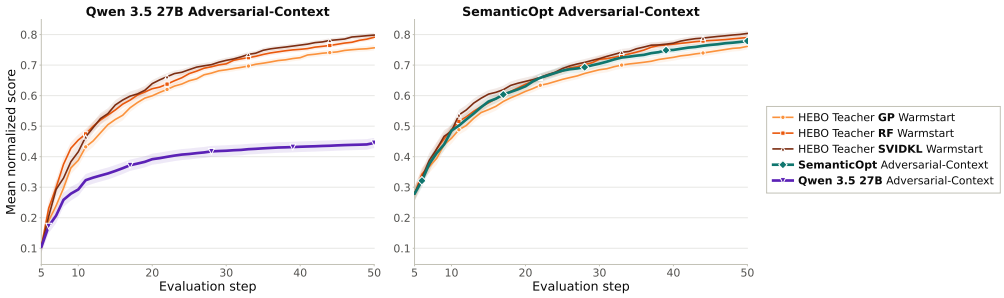
as high as SemanticOpt alone. SemanticOpt reaches teacher-level refinement within a single model, without a separate optimizer or a chosen point at which to hand off, and it also states a calibrated predictive distribution for each proposal (Figure 9). Initializing a numerical optimizer with SemanticOpt's proposals is a natural combination of the two.



(a) Full context, ten seeds per benchmark.



(b) Counterfactual context, five seeds per benchmark.



(c) Adversarial context, five seeds per benchmark.

Figure 17: Absolute trajectories behind Figure 8. Each panel starts from five configurations proposed by the LLM, then continues with that LLM or with each HEBO teacher. We see that SemanticOpt performs iterative optimization similar to the BO teachers, whereas Qwen 3.5 27B fails to match the performance. Curves show the mean and standard error over seeds.